Title	ソースコードモジュール重要度算出法の提案 (理論計算機科学の新展開)
Author(s)	後藤, 隆彰; 山田, 節夫; 西野, 哲朗; 土田, 賢省
Citation	数理解析研究所講究録 (2013), 1849: 22-27
Issue Date	2013-08
URL	http://hdl.handle.net/2433/195115
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
第1849巻 2013年 22-27

ソースコードモジュール重要度算出法の提案

Method of calculating importance for source code module

後藤 隆彰
電気通信大学大学院
情報理工学研究科
Takaaki Goto
Graduate School of
Informatics and Engineering,
The University of Electro-Communications

山田 節夫
日本電信電話株式会社
Setsuo Yamada
Nippon Telegraph and
Telephone Corporation

西野 哲朗
電気通信大学大学院
情報理工学研究科
Tetsuro Nishino
Graduate School of
Informatics and Engineering,
The University of Electro-Communications

土田 賢省
東洋大学
総合情報学部
Kensei Tsuchida
Undergraduate School of
Information Sciences and Arts,
Toyo University

1 はじめに

近年，ソフトウェア開発の短期化，大規模化が進み，仕様書やマニュアル等の膨大なソフトウェアドキュメントの作成・管理が大問題となっている．さらに，オープンソース・ソフトウェア（OSS）を活用した開発が広まりつつあるがOSSには十分なドキュメントが付されていない場合が多く，開発者に十分な情報が与えられていないのが現状である．そのため，ソフトウェアのメンテナンスやOSSを利用した開発を行う場合には，ソースコードを読んで理解することが必要となる．

一方，OSSやフリーソフトウェアの普及からソフトウェアのソースコードが手に入る機会が依然よりも増し，OSSを改変する目的やコーディングスキル向上の目的で，ソースコードリーディングを行い，ソフトウェアの仕組みや挙動を静的・動的に解析する機会も増加している．しかしながら，OSSのドキュメントが不足し，ソフトウェアの仕様を理解するために必要な情報が十分でない場合や，規模が大きいシステムを解析する際に，ソースコードをどこから解析していくか判別しにくい状況が考えられる．

これまでに，ソースコードの要約に関する研究[1, 2, 3]や，ソースコードマイニングに関する研究が行われている[4, 5, 6]．ソースコードを読む際に，ソースコードを要約したドキュメントがあると，ソースコード全体を読まずに概要を把握することができる．一方で，要約で全体像を把握した後に，実際にソースコードを改変する必要がある場合には，ソースコードを読み進める順序のガイドが提示されることも重要であると考えられる．そこで本研究では，ソースコードリーディングを支援するための，ソースコー

ドのモジュール重要度を求める方法を提案する．具体的には Java 言語を対象としてクラスの依存関係から重要度を算出する．

2 ソースコードリーディング支援環境

ソースコードの規模の大小に関わらず，ソースコードリーディングを行う場合には読み手を支援することが重要である．ソースコードリーディングを行う方法にはいくつかの方法が考えられる [7]．1つ目は，着目する機能が実装されている箇所をキーワード検索で探し，該当箇所を直接読む方法である．2つ目は，ソースコードを main 部から順次読み進めていく方法である．ソースコードの全体の流れを把握したい場合などで用いられる方法である．3つ目は，デバッガを用いて行う方法で，着目すべき箇所にブレークポイントを設定してプログラムをデバッグモードで実行し，ブレークポイントを設定した箇所に処理が進んだ際に，変数が保持する値を参照しながらソースコードを読む方法である．

本研究では，ソースコードモジュールの重要度に基づいてソースコードリーディングを支援するシステムの構築を目指す．図 1 はシステムの構成図である．本システムの入力はソースコード，出力はソースコードのドキュメントである．

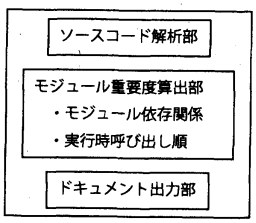

図 1: システム構成

ソースコード解析部では，入力されたソースコードの字句解析，構文解析を行う．モジュール重要度算出部では，ドキュメント出力の際に使用する，ソースコード中の各モジュールの重要度を算出する．ドキュメント出力部では，得られたモジュールの重要度に基づいて並べられたソースコードのドキュメントが出力される．

モジュール重要度の算出では，ソースコード内の各モジュールのうちどのモジュールが重要かを，モジュールの依存関係から求める方法と，プログラム実行順から求める方法の 2 つを対象とする．本稿では，モジュール依存関係からのモジュール重要度を算出する方法について述べる．

3 ソースコードモジュール重要度

ここでは，ソースコードから得られる情報に基づき重要度を定める方法を提案する．本研究では，Java 言語を用い，クラスの依存関係を対象とする．

重要度は，ソースコード中の文字列を解析し，クラス間の参照，被参照の関係を求める．参照，被参照の合計数が大きいクラスはプログラム中でよく使われているクラスであり，ソースコード中で重要なクラスと考えられる．具体的な処理の流れは以下のようになる．

1. 入力されたソースコードを字句解析し，クラスとインスタンスに関するトークンの情報を収集する．
2. 得られた情報から，クラスの参照，被参照クラスの数を集計する．
3. クラスの参照，被参照クラスの和の大きいクラスから降順にソートする．ソートを行う場合には，以下のルールで行う．
   (a) 参照，被参照，の合計が同じ場合は，被参照数が大きいクラスの順に並べる．
   (b) 参照，被参照，の合計が同じ場合で，被参照数も同じ場合は，参照数が大きいクラスの順に並べる．
   (c) 参照，被参照，の合計が同じ場合で，参照，被参照それぞれの値が同じ場合は，参照元，

参照先のクラスの重要度の最大値が最も大きいクラスの順に並べる.

(d) 隣接するクラスの重要度の最大値も同じ場合は，どちらを優先しても良い.

(e) 上記のルールで決まらない場合は，呼び出し関係の上位にあるものを優先とする.

4 ケーススタディ

ここでは，小規模な Java のソースコードを対象にケーススタディを行う．対象とするソフトウェアは，電気通信大学のソフトウェア開発の演習用で用いている，画像処理プログラムの雛形プログラム SimColorBase である [8]．SimColorBase のスクリーンショットを図 2 に示す.

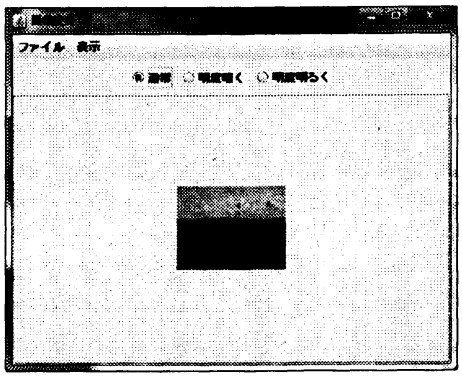

図 2: SimColorBase のスクリーンショット

このソフトウェアは，“dyschromatopsia” と “dyschromatopsia.filter” という 2 つのパッケージから構成されており，“dyschromatopsia” パッケージには，“ImageFileChooserFilter.java”，“ImageOpenFile.java”，“ImagePanel.java”，“SimWindow.java”，の 4 つのクラスが含まれている．“dyschromatopsia.filter” パッケージには，“BrighterFilter.java”，“DarkerFilter.java” の 2 つのクラスが含まれている.

ImagePanel クラスでは，「通常」，「明度暗く」，「明度明るく」の画像変換の処理を行うためのラジオボタンを格納するパネルの作成やラジオボタンを選択した場合の処理内容が記述されており，このプログラムの核となるクラスである．SimWindow クラスは，main メソッドを含むクラスであり，本アプリケーションは SimWindow クラスを起点に動作する．また，SimWindow クラスは，図 2 の画面上部にあるメニューバーの設定や，メニューバーに含まれるファイル操作の処理を含んでいる.

一方，具体的な画像のフィルタ処理を行っているのは，「明度暗く」の処理の場合は，DarkFilter クラス，「明度明るく」の処理の場合は，BrighterFilter クラス，である.

SimWindow クラスからは ImagePanel クラスと ImageOpenFile クラスのインスタンスが生成されているため，SimWindow クラスから ImagePanel クラスを参照し，同様に，SimWindow クラスから ImageOpenFile クラスを参照している．このように，ソースコードを解析し，クラス間の参照，被参照の関係を集計してルールに従いソーティングしたものが表 1 である.

表 1 より，ImagePanel クラスが被参照クラスと参照クラスの合計の数が最も多く，SimWindow クラスから ImageFileFilter クラスと続く．この順番に従い，ソースコードまたはドキュメントを提示する.

各クラス間の参照関係を基に作成したクラス間の依存関係は図 3 のようになる．この図は，ecliplse 4.2 にグラフベースの可視化・解析プラグインである ispace [9] をインストールした環境上で生成している．このように，クラスの関係をグラフ構造で描画することにより，着目するクラスとそのクラスのソースコード全体における位置も直感的に把握できる.

今回のケーススタディでは，小規模のソフトウェアを対象に分析を行ったが，規模の大きなソフトウェアの場合，グラフ構造の規模も大きくなる．例として，Java の単体テストを行うためのオープンソース・ソフトウェアである JUnit [10] のバージョン 4.10 のクラス依存図を付録の図 4 に示す．このように，グラフ構造の規模が大きい場合，本研究で提案している重要度に基づいたソースコードの解析は，開発者

表 1: SimColorBase のクラス依存関係表

重要度	クラス名	被参照クラス	参照クラス	計
1	ImagePanel	1	2	3
2	ImageOpenFile	1	1	2
3	SimWindow	0	2	2
4	BrighterFileFilter	1	0	1
5	DarkerFileFilter	1	0	1
6	ImageFileChooserFilter	1	0	1

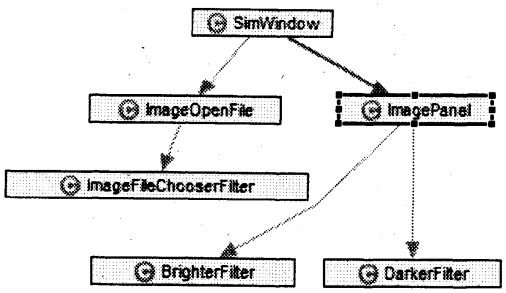

図 3: SimColorBase のクラス依存関係

への負担軽減の方法の一つとなると考えられる.

本稿では，クラスの依存関係を基に重要度の算出をしたが，別の重要度の案として，自然言語処理で用いられている tf-idf 法を応用した方法も検討している.

5 まとめ

本稿では，ソースコードリーディング支援における，ソースコード提示順を決めるためのモジュール重要度の提案を行った. さらに，ケーススタディとして小規模なソフトウェアに対する手法の実証を行った. 今後の課題としては，より規模の大きなソフトウェアに対する適用が挙げられる. さらに，提案手法を実装してソースコードリーディング支援環境の構築を構築し，ソフトウェア開発者による評価も行いたい.

ソースコードリーディング支援環境では，本稿で提案した「どの順序でソースコードを読むか」という観点の支援の他にも,「ソースコード全体の中でどこの箇所を読んでいるか」という観点の支援も必要であると考えられる. ソースコードリーディング中に全体像と着目点の位置関係を理解させるために，シーケンス図等の図式表現と連動した方法についても検討していきたい.

参考文献

[1] Sonia Haiduc, Jairo Aponte, and Andrian Marcus, Supporting program comprehension with source code summarization, *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pp. 223–226, New York, NY, USA, 2010. ACM.

[2] S. Haiduc, J. Aponte, L. Moreno, and A. Marcus, On the use of automated text summarization techniques for summarizing source code, *Proceedings of 2010 17th Working Conference on Reverse Engineering (WCRE)*, pp. 35 –44, oct. 2010.

[3] Sarah Rastkar, Gail C. Murphy, and Alexander W.J. Bradley, Generating natural language summaries for crosscutting source code

concerns, *Proceedings of 2011 27th IEEE International Conference on Software Maintenance (ICSM)*, pp. 103 –112, sept. 2011.

[4] D. Poshyvanyk, A. Marcus, and Y. Dong, Jiriss - an eclipse plug-in for source code exploration, *Proceedings of 14th IEEE International Conference on Program Comprehension, 2006. ICPC 2006*, pp. 252 –255, 2006.

[5] E. Enslen, E. Hill, L. Pollock, and K. Vijay-Shanker, Mining source code to automatically split identifiers for software analysis, *Proceedings of 6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR '09*, pp. 71 –80, may 2009.

[6] 小林隆志, 林晋平, データマイニング技術を応用したソフトウェア構築・保守支援の研究動向, コンピュータ ソフトウェア, Vol. 27, No. 3, pp. 3_13–3_23, 2010.

[7] まつもとゆきひろ. オープンソース/C 言語に学ぶ「ソースコードの読み方」. 日経ソフトウェア 2007 年 1 月号. 日経 BP 社, 2007.

[8] UEC ソフトウェア・リポジトリ, プログラム開発教材 色弱者支援画像変換処理プログラム. `https://www.repository.uec.ac.jp/`.

[9] ispace. `http://ispace.stribor.de/index.php?n=Ispace.Home`.

[10] Junit. `http://junit.sourceforge.net/`.

付録 A　JUnit のクラス依存図

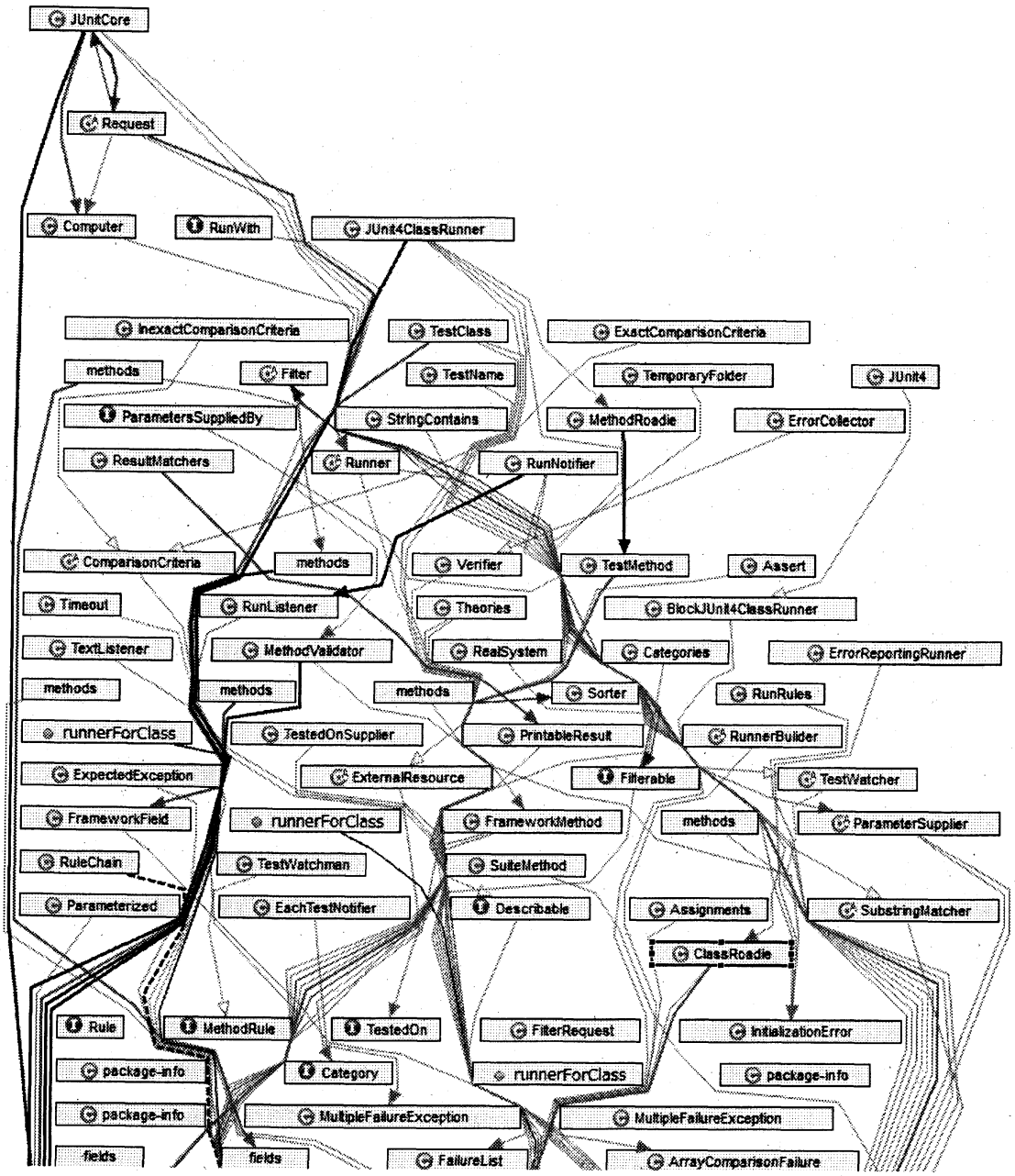

図 4: JUnit のクラス依存関係